

Los comentarios son míos, resumen la corrección.  
Los 4 ejercicios están aprobados.

1

$$a) G(i, v) = \begin{cases} \min \{ G(i+1, v-k+v_i) + t_{ik} : 0 \leq k \leq \min(5, v) \} \\ 0 \quad \text{si } i > n \end{cases}$$

$\text{si } i \leq n$

Siendo:

$K$ : la cantidad de votos que gasta en ese nivel

$v_i$ : la cantidad de votos que puede ganar en ese nivel (enunciado)

faltaba dar más explicación

b) En el peor caso, cada subproblema llama a 5 subproblemas, entonces tenemos  $O(5^n)$ . La cantidad de subproblemas diferentes son  $n \cdot n^2$  es decir  $O(n^3)$ . Pero pensándolo un poco mejor podemos ver que nunca vamos a gastar más de  $5n$  votos en total (porque solo podemos usar como máximo 5 votos en cada nivel). Entonces tenemos  $n^2$  subproblemas diferentes.

$$n^2 \ll 5^n \quad \forall x$$

c) ~~se~~ Utilizamos una matriz PD de  $n \times 5n$ , donde vamos a guardar los ~~se~~ resultados de los subproblemas.

~~se~~ Resuelvo recursivamente  $G(i, v)$  como:

> si  $i > n$  devuelvo 0

> sino devuelvo el mínimo entre:  $G(i+1, v-k+v_i)$

$$G(i+1, \min(5n, v-k+v_i)) + t_{ik}$$

con  $k$  yendo de 0 a  $\min(5, v)$  (osea hacer un for)

> Siempre después de calcular el valor y antes de retornarlo lo guardamos en la matriz PD en la posición  $(i, v)$ . ~~se~~ Y también antes de calcularlo se tiene que fijar si lo calculó antes.



d)  $G(i, v)$ :

if  $DP[i][v] \neq \perp$ :

return  $DP[i][v]$ ;

$m \leftarrow \text{INF}$

for  $k$  desde 0 a  $\min(S, v)$ :

$m \leftarrow \min(m, G(i+1, \min(S, v-k+v_i)) + t_{ik})$ ;

fin para

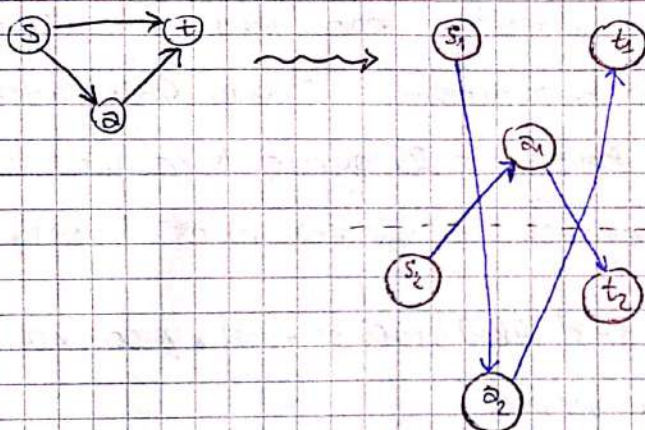
$DP[i][v] = m$

return  $m$ ;



(2) ~~Vamos~~ Vamos a modelar de tal forma que vamos a tener dos niveles en el grafo, uno para cuando estamos a distancia par y otro para impar. En ambos niveles vamos a tener los mismos vértices que tiene  $G$ . Las aristas las borramos todas ~~y solo~~ y vamos a agregar las aristas de la siguiente manera. Si en  $G$  existe una arista  $e = (v, w)$  entonces en este nuevo grafo vamos a agregar  $e' = (v_1, w_2)$  y  $e'' = (v_2, w_1)$  para todas las aristas de  $G$ . Con este modelo simplemente usamos BFS desde  $s_1$  y queremos llegar a  $t_1$ .

Ejemplo:



El nivel 1 representa cuando estamos a distancia par, y solo podemos estar en esos vértices cuando recorramos un camino par. El nivel 2 representa cuando estamos a distancia impar y solamente podemos llegar a esos nodos cuando recorramos un camino impar.

Realizando BFS ~~esto~~ y guardándonos el antecesor podemos guardar el camino y luego para el camino final simplemente removemos de que nivel era.

Es decir, si el recorrido final es:  $s_1, a_2, t_1$  lo transformamos a:  $s, a, t$ .



③ ~~Para~~ Supongamos que el árbol es fácil.

I) Realizo DFS desde un nodo cualquiera y armamos un árbol. ~~Si~~ Se puede llegar.

II) Cuando tenemos el árbol armado, por cada arista que no usamos ~~se~~ ~~(v → w)~~ nos fijamos en el camino del árbol entre  $v$  y  $w$  (es único porque es un árbol) si hay una arista que peso más. Si eso pasa entonces sacamos la máxima arista del ciclo y colocamos  $(v → w)$  en el árbol. ~~Esto sirve porque las aristas como el árbol al aumentar~~

Para encontrar el ciclo simplemente es entre  $v$  y  $w$  subir hasta un ancestro en común. **No es el ancestro común**

~~Se~~ Vamos a tener un contador por cada arista que vamos a aumentar cuando lo recorramos cuando observamos el ciclo. Si el contador ~~allega~~ llega a 2 quiere decir que esa arista pertenece a 2 ciclos, entonces el grafo no es fácil.

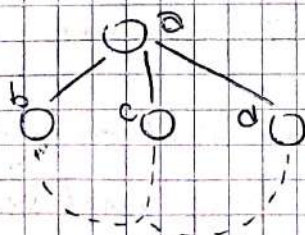
Veamos que  $m < 3 \cdot n$  si el ~~grafo~~ grafo es fácil y conexo.

Por inducción en la cant de vértices:

CB: 0  $0 < 3$  ✓ ~~PI~~

PI: Tenemos un grafo fácil y conexo. y agregamos un nodo, el nodo lo tenemos que conectar para que el grafo sea conexo.

Si lo conectamos a ~~3~~ 3 o más vértices, entonces:



La inducción asume algo no citado: el como se puede construir un grafo fácil.

Sea  $a$  el nodo q agregamos, aparecen los 2 ciclos: ~~P~~

$$\left. \begin{array}{l} > (a \rightarrow b) + P_{bc} + (c \rightarrow a) \\ > (a \rightarrow c) + P_{cd} + (d \rightarrow a) \end{array} \right\} \text{Existen porque es conexo.}$$

Abst! Entonces solo podemos agregar 2 vértices cuando agregamos un nodo.



✱

Entonces:  $m < 3n$  (\*)Entonces si primero podemos filtrar los grafos que  $|E(G)| \geq |V(G)|$ .

En

Complejidad:

I) DFS es  $O(m+n) = O(n)$ II) Como tenemos el contador vemos que solo pasamos una vez por cada arista, entonces  $O(m) = O(n)$ Finalmente, tenemos en total  $O(m+n) = O(n)$ 

(\*) Esto no quiere decir que un nodo no pueda tener más de 2 aristas. Solamente demostramos que hay una cota para las aristas.



④ I) Realizamos Dijkstra desde  $s$ . Entonces obtenemos  $d(s, v) \forall v$ .

II) Damos vueltas las aristas del grafo y calculamos Dijkstra desde  $t$ . Entonces obtenemos  $d(v, t) \forall v$ .

III) Obtenemos el conjunto de aristas y las ordenamos por peso decreciente. Por cada una calculamos (sea  $(v \rightarrow w)$  la arista actual):

$$Z = d(s, v) + d((v \rightarrow w)) + d(w, t)$$

Si  $Z \leq C$  entonces nos quedamos con esa arista y es la solución y el peso del camino es  $Z$ .

$Z$  es el peso del camino mínimo entre  $s$  y  $t$  que pasa por  $(v \rightarrow w)$ .

>  $d(s, w)$  lo calculamos en I.

>  $d(v \rightarrow w)$  es dato.

>  $d(w, t)$  lo calculamos en II.

Complejidad: ① lo podemos hacer en  $O(\min(n^2, m \log n))$

② Por vuelta las aristas es  $O(m)$ . Luego  $\Rightarrow$  igual que ①.

③ Ordenar en  $O(m \log m)$  e iterarlas en  $O(m)$

Final:  $O(\min(n^2, m \log n) + m \log m)$

No hacía falta ordenar los vértices, mejor búsqueda lineal